

GridCover - Autonomous Lawn Mower ("cutter") Simulation

Overview

GridCover is a simulation program that physically models how an autonomous lawn mower, or "cutter", cuts grass in a rectangular area with optionally placed obstacles. The simulation tracks the cutter as it moves across the area, bouncing off boundaries and specified obstacles and changing direction according to the defined strategy to achieve complete grass coverage.

The simulation is a close physical model of the cutting mechanism of the two most common types of robotic cutter. Currently the bounding area is a rectangle where an optional obstacle-map can be added with a user-specified number of obstacles. Obstacles can be shaped like rectangles, arbitrary polygons, circles or line with a thickness. The simulation is accurate down to a chosen cell size (usually somewhere around 1cm) and hence will very accurately simulate real time. The performance is such that it is possible to simulate a 100x100m rectangle with 1cm cell size in slightly less than 3min on a desktop computer.

The completed simulation can be shown as a animated video with a user specified fps or as an image of the completed simulation showing the paths of the cutter and obstacles (or both).

Main features

- **Simulation based on accurate physics**, with configurable cutter details like mower radius, speed, and direction perturbation
- **Multiple stopping conditions**, bounces, time, coverage percentage, distance, or simulation steps
- **Visual output** with color-coded PNG images showing coverage patterns or as a animated video of the entire simulation
- **Reproducible results** using random seeds for deterministic simulations
- **Flexible configuration** with extensive command-line options (see below)
- **SQLite Database support** Option to store simulation results in a SQLite DB
- **HW Assisted animation encoding** Creation of animated video of simulation with support for HW encoding

The model is physically accurate to a degree where it is meaningful to use it as a base for statistical investigations that are quite hard with an analytical approach. Such as the question *"How does distance traveled correspond to area covered?"* or *"How does the first derivative of distance change with increasing coverage?"* (which can be used to illustrate the law of diminishing returns).

Installation

MacOS

The distribution have pre-built packages for both Apple silicon and Intel architecture. These packages can be installed either from *Finder* by double-clicking on the file or the command line with:

```
installer -pkg <pkg-name>
```

by default the program will be installed in `/usr/local/bin`

Linux

The distribution have a RPM package for Fedora/Red Hat/CentOs. Install with:

```
sudo dnf install gridcover-2.1.0-1.fc42.x86_64.rpm
```

Windows

See the user guide for installation from source

Command Line Options

Below is a categorized list of all command line options for **gridcover** to help to find and understand the available configuration parameters.

Simulation Area & Grid

- `-W, --grid-width <GRID_WIDTH>` Width in units of the grid
- `-H, --grid-height <GRID_HEIGHT>` Height in units of the grid
- `-s, --square-size <SQUARE_SIZE>` Size of each grid square
- `-M, --map-file-name <MAP-FILE>` Path to map file with obstacles

Cutter & Physics

- `-r, --radius <RADIUS>` Radius of the cutter plate
- `-l, --blade-len <BLADE_LEN>` Length of knife blade
- `-T, --cutter-type <CUTTER-TYPE>` Cutter type: 'blade', 'circular'
- `-v, --velocity <VELOCITY>` Movement velocity in units/second
- `-x, --start-x <START_X>` Starting X coordinate for the cutter, random if not specified
- `-y, --start-y <START_Y>` Starting Y coordinate for the cutter, random if not specified
- `--dir-x <DIR_X>` Start direction X component, random if not specified
- `--dir-y <DIR_Y>` Start direction Y component, random if not specified
- `-p, --perturb <True/False>` Use perturbation angle for direction changes at edge bound
- `-k, --perturb-segment <True/False>` Use perturbation randomly while moving in a straight line
- `--perturb-segment-percent <PERTURB_SEGMENT_PERCENT>` Perturb segment percent chance per cell travelled
- `-C, --track-center <True/False>` Turn visual centerpoint tracking on or off in the output image

Wheel slippage

- `--wheel-slippage <True/False>` Enable wheel slippage
- `--slippage-probability <PROBABILITY>` Probability slippage may happen during an activation distance
- `--slippage-min-distance <LENGTH>` Minimums distance slippage will be enabled for
- `--slippage-max_distance <LENGTH>` Maximum distance slippage will be enabled for
- `--slippage-radius-min <RADIUS>` The minimum radius to model turning
- `--slippage-radius-max <RADIUS>` The maximum radius to model turning
- `--slippage-check-activation-distance <LENGTH>` Interval when we randomly check if there is a slippage
- `--slippage-adjustment-step <LENGTH>` Step size for wheel slippage adjustment

Wheel inbalance

- `--wheel-inbalance <True/False>` Enable wheel inbalance
- `--wheel-inbalance-radius-min <RADIUS>` Minimum turning radius to model inbalance
- `--wheel-inbalance-radius-max <RADIUS>` Maximum turning radius to model inbalance
- `--wheel-inbalance-adjustment-step` Step size for wheel inbalance adjustment

Simulation Control & Stopping Conditions

- `-z, --step-size <STEP_SIZE>` Simulation step size in units, automatically determined if not specified
- `-b, --stop-bounces <STOP_BOUNCES>` Maximum number of bounces before ending simulation
- `-t, --stop-time <STOP_TIME>` Maximum simulated time when to stop (seconds)
- `-c, --stop-coverage <STOP_COVERAGE>` Stop when reaching this coverage percentage
- `-m, --stop-simsteps <STOP_SIMSTEPS>` Stop after specified number of simulation steps
- `-d, --stop-distance <STOP_DISTANCE>` Stop after specified distance covered

Battery & Charging

- `-B, --battery-run-time <BATTERY_RUN_TIME>` Battery duration in minutes
- `-A, --battery-charge-time <BATTERY_CHARGE_TIME>` Battery charging time in minutes

Output & Visualization

- `-o <IMAGE-FILE-NAME>` Output image file name
- `--image-width-mm <IMAGE_WIDTH_MM>` Image output width in mm
- `--image-height-mm <IMAGE_HEIGHT_MM>` Image output height in mm
- `-Z, --paper-size <PAPER-SIZE>` Paper size for output image
- `-D, --dpi <DPI>` DPI setting for image output
- `-G, --show-gridlines <True/False>` Show or hide gridlines in output image
- `--color-theme <COLOR-THEME>` Color theme for output image, can be one of: 'default', 'green30', 'blue', 'orange_red', 'gray_green', 'pure_green'.
- `show-quad-tree <True/False>` Shows the spatial index that is built as a quad-tree in the image
- `show_image_label <True/False>` Shows the image label with sim-time and coverage in the top-left corner of the image

Animation & Video

- `-f, --generate-frames <True/False>` Generate frames for an animation
- `-F, --frame-rate <FRAME_RATE>` Specify frame-rate for the animation
- `--frames-dir <FRAMES-DIR>` Directory for animation frames
- `-a, --create-animation <True/False>` Generate an animation video from the frames (implies `-f` if not set)
- `--animation-file-name <ANIMATION-FILE-NAME>` Animation file name
- `--hw-encoding <True/False>` Use HW assisted encoding for animation (macOS/Linux)
- `--delete-frames <True/False>` Delete frames after animation has been created
- `-U, --animation-speedup` Speedup factor for the animation video, this makes the real time go x-times faster
- `--start-threaded-encoding` Experimental encoding that is done asynchronous with the simulation

Output Formatting & Reporting

- `-J, --json-output <True/False>` Print result of simulation as a JSON object
- `--verbosity <VERBOSITY>` Verbosity during simulation, 0 (default), 1, 2
- `-R, --show-progress <True/False>` Show progress bar during simulation
- `-q, --quiet <True/False>` Quiet, no output at all
- `-X, --generate_json_files <True/False>` Regardless of other settings generate "model.json" and "result.json" in current directory.

Randomness & Reproducibility

- `-S, --random-seed <RANDOM_SEED>` Random seed for reproducible results

Performance

- `-P, --parallel <True/False>` **Deprecated.** Use parallel processing when possible to speed up simulation
- `--use-quad-tree <True/False>` Use a spatial index based on quad-trees to speed up obstacle collision detection. This give 10-20% speedup
- `--min-qnode-size <VALUE>` Set minimum quad-tree node size in multiples of cutter radius
- `--show-quad-tree <True/False>` Show quad-tree in the output image
- `--save-quad-tree <True/False>` Save quad-tree to file with the same name as the map-file but with "_index.json" appended to the basename

Configuration Files & Database

- `--args-write-file-name <ARGS-FILE-NAME>` Write program arguments file in TOML format
- `-i, --args-read-file-name <ARGS-FILE-NAME>` Read program arguments from a TOML file
- `-Q, --database-file <DATABASE-FILE>` Store simulation results and model parameters in SQLite database file
- `--generate-completions` Generate autocompletion files for `bash` and `zsh`

Help & Version

- `-h, --help` Print help
- `-V, --version` Print version

Shell autocompletions

As there are quite a few options it is recommended to install the shell autocompletion extensions for `gridcover`. They can be either manually installed from `assets/completions/`, both `bash` and `zsh` are supported. They can also be automatically installed by running either one of the two makefile targets `make bash-install` or `make zsh-install`.

If the autocompletion files are missing they can be automatically generated by running `gridcover --generate-completions`. This will generate completion files for both `bash` and `zsh` in the current directory as `gridcover.bash` and `_gridcover` following the naming conditions for `bash` and `zsh` respectively.

Color Themes

gridcover supports multiple color themes for visualizing the coverage results. Color themes define:

- Grid background color
- Obstacle color
- Center point tracking color
- Coverage intensity colors (gradient from light to dark based on visit frequency)

Available Themes

- **default** - Green coverage gradient with gray background (21 shades)
- **green30** - Extended green gradient . similar to default but finer steps (30 shades)
- **blue** - Blue coverage gradient for alternative visualization
- **orange_red** - High contrast 10 color gradient
- **gray_green** - A more subtle green 30 color gradient
- **pure_green** - A vivid green 30 color gradient

Using Color Themes

Color themes are automatically applied when generating images or animations. The default theme is used unless specified otherwise as a command line argument `--color-theme`

Adding Custom Themes

To add a new color theme, create a new `ColorTheme` struct and register it with the `ColorThemeManager`:

```
use crate::color_theme::{ColorTheme, ColorThemeManager};

let custom_theme = ColorTheme {
    name: "custom".to_string(),
```

```

    grid_background_color: [255, 255, 255], // White background
    grid_line_color: [128, 128, 128],      // Gray lines
    obstacle_color: [255, 0, 0],           // Red obstacles
    center_color: [0, 0, 0],               // Black center points
    coverage_shades: vec![
        [255, 255, 0], // Yellow (light coverage)
        [255, 128, 0], // Orange
        [255, 0, 0],   // Red (heavy coverage)
        // Add more colors as needed...
    ],
};

let mut theme_manager = ColorThemeManager::new();
theme_manager.register_theme(custom_theme);

```

Examples

1. Basic usage

Use default values and simulate covering 50% of the default rectangle size (10x10 units). As long as all values are consistent the default units can be thought of to be cm, dm, m etc. However, the default values are set so it makes physical sense thinking of units as meters.

This, the most basic of all examples, answers the question *"How long time does it take to cut X % of the lawn"* It is important to realize that the result is just one possible outcome from the distribution of possible outcomes. Running the same example one more time will give another outcome from the distribution.

Note: By specifying the random seed (with `-S`) the exact same result can be achieved no matter how many time the example is run.

```

$> gridcover -R true -c 50
Coverage: 50.00% ( 70317/ 140625 cells covered), Distance: 146.64,
Bounces: 29, Sim-Time: 00:08:08

Result
=====
Coverage
  Coverage.Bounces           : 29
  Coverage.Distance         : 146.6400000000051
  Coverage.Percent          : 50.00320000000001
Cutter
  Battery
    Cutter.Battery.Charge count : 0
    Cutter.Battery.Charge left (%) : 100.0
  Cutter.Blade Length         : 0.05
  Cutter.Radius               : 0.15
  Cutter.Type                 : "blade"

```

```

Cutter.Velocity          : 0.3
Time
  Time.CPU               : "00:00:00.442"
  Time.Cutting           : "00:08:08"

```

From the result we can see that it took 6min and 56s to cut 50% of the area.

2. Basic usage with generated path image

If we change the first example in two ways:

1. We want to see how much coverage we get if we let the cutter run, say 500 m, using the `-d` option
2. We want to see a picture of the simulation with the path of the cutter using the `-o` option

```

$> gridcover -R true -d 500 -o test.png
Coverage: 89.65% ( 126064/ 140625 cells covered), Distance: 500.00,
Bounces: 106, Sim-Time: 00:27:46

```

```

Result
=====
Coverage
  Coverage.Bounces          : 106
  Coverage.Distance         : 500.00400000000185
  Coverage.Percent          : 89.64551111111111
Cutter
  Battery
    Cutter.Battery.Charge count : 0
    Cutter.Battery.Charge left (%) : 100.0
  Cutter.Blade Length       : 0.05
  Cutter.Radius              : 0.15
  Cutter.Type                : "blade"
  Cutter.Velocity            : 0.3
Time
  Time.CPU                   : "00:00:01.708"
  Time.Cutting               : "00:27:46"

```

Note: The efficiency is not calculated when the stopping condition is distance travelled as it is not applicable (as it is only possible to calculate a minimum time for coverage)

3. Basic usage with simulated battery capacity limit

One way to increase the fidelity of the simulation is to add the realism that a battery with finite capacity is being used and that needs regular re-charging. This will add to the simulation time (i.e. the time it takes to cut lawn until the selected stop condition is met). There are two times that needs specifying

1. How long the battery lasts (in min) using the option `-B`
2. How long time it takes to charge the battery (in min) using the option `-A`

In addition a small time (1-15min) is randomly added (automatically) to simulate the time it takes for the cutter to move back to the charging station.

Note: We increased the stopping distance so we run out of battery to see the result of the battery charging.

```
$> gridcover -R true -d 2000 -B 90 -A 180 -o test.png
Coverage: 98.57% ( 246435/ 250000 cells covered), Distance: 2000.00,
Bounces: 320, Sim-Time: 04:58:13, Battery capacity left: 76.5%

Result
=====
Coverage
  Coverage.Bounces           : 407
  Coverage.Distance         : 2000.0039999930268
  Coverage.Percent          : 99.2256
Cutter
  Battery
    Cutter.Battery.Charge count      : 1
    Cutter.Battery.Charge left (%)   : 76.5437037037143
  Cutter.Blade Length           : 0.05
  Cutter.Radius                 : 0.15
  Cutter.Type                   : "blade"
  Cutter.Velocity               : 0.3
Time
  Time.CPU                     : "00:00:07.391"
  Time.Cutting                  : "04:54:51"
```

4. Using obstacle map and animation video

In this example we:

1. load a basic obstacle map and generate an animation video (using **-M**)
2. add a higher level of verbosity (using **--verbosity 1**) which means that the output will give more detailed information about the simulation.
3. add a smaller cell size (using **-s 0.01**)
4. add an animation video (using **-a true**)

```
$> gridcover -M assets/mapex01.yaml -s 0.01 -a true -R true -c 75 --
verbosity 1
Frame: 006761, Coverage: 75.00% ( 706408/ 941859 cells covered), Distance:
405.68, Bounces: 86, Sim-Time: 00:22:32
Video created successfully: cutter_sim.mp4

Simulation Result
=====
Coverage
  Coverage.Bounces           : 152
  Coverage.Cells              : 706403
```

```

Coverage.Max visited      : 19
Coverage.Min visited      : 1
Coverage.Percent          : 75.00092901379081
Cutter
  Battery
    Cutter.Battery.Charge count      : 0
    Cutter.Battery.Charge left (%)   : 100.0
    Cutter.Battery.Charge time       : 120.0
    Cutter.Battery.Run time          : 0.0
  Cutter.Blade Length        : 0.05
  Cutter.Cells under         : 1600.0
  Cutter.Distance           : 424.81799999921617
  Cutter.Radius             : 0.2
  Cutter.Type               : "blade"
  Cutter.Velocity           : 0.3
Frames
  Frames.Animation          : true
  Frames.Animation file name : "cutter_sim.mp4"
  Frames.Delete frames      : false
  Frames.Directory          : "frames_dir"
  Frames.Enabled            : true
  Frames.HW Encoding        : true
  Frames.Rate (fps)         : 5
  Frames.Steps per frame    : 10
Grid
  Grid.Cell side (units)    : 0.01
  Grid.Height (units)       : 10.0
  Grid.Hor.Cells            : 1000
  Obstacles
    Grid.Obstacles.NumCells : 58141
    Grid.Obstacles.Percent  : 5.8141
  Grid.Total cells         : 1000000
  Grid.Vert.Cells          : 1000
  Grid.Width (units)       : 10.0
Output image
  Output image.DPI          : 300
  Output image.File name    : ""
  Paper size
    Output image.Paper size.format : "A4"
    Output image.Paper size.height_mm : 297.0
    Output image.Paper size.width_mm : 210.0
  Pixels
    Output image.Pixels.height : 3508
    Output image.Pixels.width  : 2480
  Output image.Show gridlines : false
Start
  Start.Angle (degrees)     : 244.31604360192992
  Direction
    Start.Direction.X       : -0.43340675369924286
    Start.Direction.Y       : -0.9011984164699158
  Position
    Start.Position.X        : 8.093621061635275
    Start.Position.Y        : 4.600970808726622
Steps

```

```

Steps.Length (units)           : 0.006
Steps.Seconds/step             : 0.02
Steps.Steps/cell               : 0.6
Steps.Steps/second             : 50
Steps.Total #                  : 70803
Time
Time.CPU time                  : "00:02:39.316"
Time.Cutting time              : "00:23:36"
Time.Cutting time (seconds)    : 1416
Time.Efficiency                : 41.52
Time.FFmpeg Encoding Duration : "00:01:42.039"
Time.Min.Cov.Time              : "00:09:48"
Time.Min.Cov.Time (seconds)    : 588

```

4. Using obstacle map and speeding up the animation video

By default the animation is a 1:1 with real time, meaning that if the cutting takes 2h then the video will be 2h. This is often not very practical and in addition the animations might get quite large. To make it both faster and smaller we can speed them up. For example, if the cutting takes 120min then if we speed up by a factor of 20 the video will be $120/20 = 6\text{min}$ long

```

$> gridcover -M assets/mapex01.yaml -S 96783 -s 0.01 -R true -c 98 -U 20 -
a true
Video created successfully: cutter_sim.mp4

Simulation Result (Short)
=====
Coverage
  Coverage.Bounces           : 638
  Coverage.Distance          : 2160.7320000083255
  Coverage.Percent           : 98.0001252841455
Cutter
  Battery
    Cutter.Battery.Charge count : 0
    Cutter.Battery.Charge left (%) : 100.0
  Cutter.Blade Length        : 0.05
  Cutter.Radius              : 0.2
  Cutter.Type                 : "blade"
  Cutter.Velocity             : 0.3
Time
  Time.CPU                   : "00:01:00.547"
  Time.Cutting                : "02:00:02"

```

We can now check the length of the resulting simulation video

```
$> ffprobe -v error -show_entries format=duration -of
default=noprint_wrappers=1:nokey=1 cutter_sim.mp4
360.200000
```

which tells is that it is roughly 360 s (or 6min) exactly what we wanted. Note: You can also find out the length with the slightly shorter command

```
ffmpeg -i cutter_sim_1.mp4 2>&1 | grep Duration
Duration: 00:06:00.20, start: 0.000000, bitrate: 2778 kb/s
```

Statistics

Language	Files	Lines	Blank	Comment	Code	Complexity	Bytes	Uloc
Rust	23	5757	511	442	4804	659	217804	0
JSON	6	1494	0	0	1494	0	63907	0
YAML	5	157	22	23	112	0	3311	0
Markdown	4	1129	215	0	914	0	46293	0
BASH	2	521	9	11	501	15	19597	0
Shell	2	415	61	69	285	45	14146	0
Makefile	1	487	54	27	406	28	22793	0
Python	1	136	21	30	85	28	5758	0
TOML	1	34	2	0	32	0	721	0
Total	45	10130	895	602	8633	775	394330	0

Estimated Cost to Develop (organic) \$259,755
Estimated Schedule Effort (organic) 8.24 months
Estimated People Required (organic) 2.80

Example Commands

```
./target/release/gridcover -M assets/maps/mapex01.yaml -o coverage.png --
show-quad-tree true -S 385925 -c 95 -R false --verbosity 1 --use-quad-tree
true -s 0.01
```

```
./target/release/gridcover -M assets/maps/mapex01.yaml -o coverage.png --
show-quad-tree true -S 385925 -c 95 -R false --verbosity 2 --use-quad-tree
```

```
false -s 0.01 | grep -e "Collision Checks" -e "CPU"
```